Question - 1
Playlist

SCORE: **50 points**

Implementation Easy

You have just made a playlist on your MP3 player with all of your songs. Some of them may be listed more than once. Your buttons are limited and you can only go down ↓ or up ↑ in the list one song at a time. The list is circular, so when you are at the end of the list, ↓ takes you to the beginning and *vice versa*.

Given the index of the song that is currently playing, determine the minimum number of button presses you will need to reach the requested next song. For example, your playlist includes *[money, wishyouwerehere, welcometothemachine, time]*. Currently *time* is playing, index *3*, and you want to play *money* next, index *0*. You can either go down *1* or up *3* to reach the song.

Function Description

Complete the function *playlist* in the editor below. The function must return the integer value of the least button presses to get to the requested song.

playlist has the following parameter(s):

songs[songs[0],...songs[n-1]]: an array of song name strings in order of the playlist

k: the index of the song currently playing

q: the name of the song you want to play next

Constraints

- $1 \le n \le 100$
- $0 \le k \le n-1$
- $1 \le |s[i]|, |q| \le 100$
- Song *q* is in the playlist.

▼ Input Format for Custom Testing

Input from stdin will be processed as follows and passed to the function.

The first line contains an integer *n*, the size of the array *songs*.
Each of the next *n* lines contains a string *songs[i]* where $0 \le i < n$.
The next line contains the integer *k*.
The next line contains the string *q*.

▼ Sample Case 0

Sample Input 0

```
4
wheniseeyouagain
```

```
borntorun
nothingelsematters
cecelia
1
cecelia
```

Sample Output 0

```
2
```

Explanation 0

You are listening to song *s[k] = s[1] = "borntorun"*. By pressing the ↓ or the ↑ button *2* times, you can reach *q = s[3] = "cecelia"*.

▼ Sample Case 1

Sample Input 1

```
4
dancinginthedark
rio
liveoak
liveoak
0
liveoak
```

Sample Output 1

```
1
```

Explanation 1

You are listening to song *s[k] = s[0] = "dancinginthedark"*. By pressing the ↑ button *1* time, you can reach *q = s[3] = "liveoak"*. Observe that we could also have pressed the ↓ button two times to switch to *s[2] = "liveoak"*, but this would not be minimal.

Question - 2
Car Inheritance

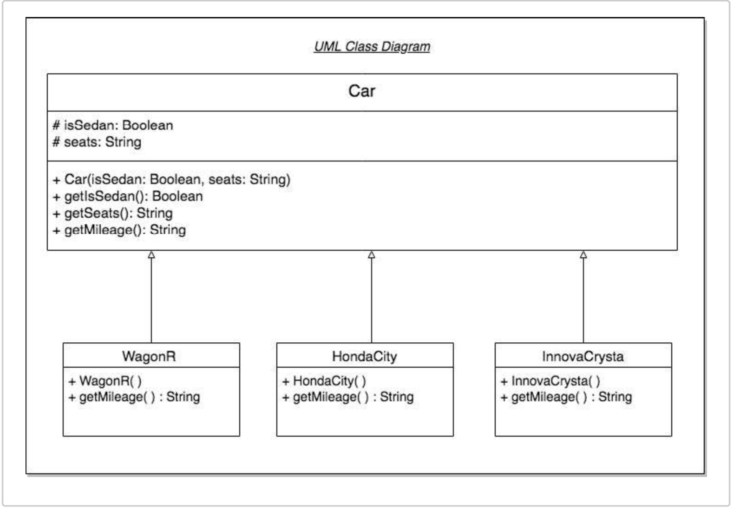
SCORE: **50 points**

Inheritance | Polymorphism | Easy | Abstraction | Java | C++ | C# | Object Oriented Programming | Java 8 | Problem Solving

In this challenge, you will be asked to build on an abstract class and initialize an instance of each class with a variable. The program will then test your implementation by retrieving the data you stored.

The locked code in the editor does the following:

1. Declares an abstract class named Car with the implementations for `getIsSedan()` and `getSeats()` methods, as well as an abstract method named `getMileage()`.
2. Creates *WagonR*, *HondaCity*, or *InnovaCrysta* object based on input (0 for WagonR, 1 for HondaCity and 2 for InnovaCrysta).
3. Calls the `getIsSedan()`, `getSeats()`, and `getMileage()` methods on the object.



The details for each car are provided below -

1. WagonR is not a sedan and has 4 seats.
2. HondaCity is a sedan and has 4 seats.
3. InnovaCrysta is not a sedan and has 6 seats.

Complete the code in the editor below to implement the following:

1. Three classes named, `WagonR`, `HondaCity`, and `InnovaCrysta` that inherit from the `Car` class.
2. One integer argument is provided to the constructor which is the mileage of the car.
3. Each class must implement the `getMileage()` method which returns a string in the form of `<mileage> kmpl` where `<mileage>` is the value provided to constructor.

Constraints

- The integer in first line will be in between 0 and 2 inclusive.
- The integer in the second line (i.e. mileage) will be in between 5 and 30 inclusive.

▼ Input Format For Custom Testing

The first line contains an integer describing the type of car to instantiate.
The second line contains an integer, the mileage of the car.

▼ Sample Case 0

Sample Input For Custom Testing

```
0
22
```

Sample Output

```
A WagonR is not Sedan, is 4-seater, and has a
mileage of around 22 kmpl.
```

▼ Sample Case 1

Sample Input For Custom Testing

```
1
12
```

Sample Output

```
A HondaCity is Sedan, is 4-seater, and has a
mileage of around 12 kmpl.
```